SCORM RTE Web Services Interface

Virginia Travers, BBN Technologies
August 11, 2008

Introduction

The SCORM Web Services interface provides a Web-Services-based interface for SCO-to-LMS communication. As defined in the SCORM 2004 3rd Edition Content Aggregation Model, an SCO is "any piece of data that can be rendered by a Web client" and "communicates with an LMS using the IEEE ECMAScript API for Content to Runtime Services Communication standard."[1] The SCORM Web Services interface described here allows *any application* to act as an SCO, removing the restriction that SCOs be displayed by a Web client.

Motivation

The SCORM standard supports browser based delivery of instructional materials. Advances in Web technology support browser delivery of rich multi-media material, including simulations. However, browser-based simulations still fall short of the fully immersive training experience that can be provided by stand-alone applications. Several projects have demonstrated the benefits of tightly integrating SCORM-conformant content and stand-alone, simulation-based training applications.[2] However, these projects were focused on single applications or specific technologies and did not provide a general framework for integrating SCORM content with stand-alone applications.

A significant limitation in these efforts was the SCORM requirement that SCOs be delivered by a Web browser and communicate through that browser to the SCORM runtime environment. This effectively prevents a stand-alone application from acting as a SCO and necessitates the use of browser-based SCOs as intermediaries between the runtime environment and the application. Furthermore, the requirement for browser based delivery introduces increasingly tight security restrictions, appropriate for protecting Web clients, but inappropriate and unnecessarily restrictive for stand-alone applications, which in fact, should include their own security measures.[3]

---

[1] IEEE 1484.11.2 Standard for Learning Technology – ECMAScript Application Programming Interface for Content to Runtime Services Communication. November 10, 2003. Available at: http://www.ieee.org/

[2] Biddle, E., Perrin, B., Dargue, B., Pike, W.Y., Marvin, D., and Lunsford, J,. "Performance- based advancement using SCORM 2004," *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference,* vol. 2006. (See also: http://www.jointadlcolab.org/downloads/research/2005/perf_based_adv/Boeing_FinalReport_A005_rev1.pdf. And see the final reports for the 2004 JADL Prototype Efforts, "Simulation-based Intelligent Training and Assessment", "Integration of HLA and SCORM", and "Integration of HLA and SCORM in a Multi-Student Context," available at http://www.jointadlcolab.org/research/2004/index.aspx.

[3] This includes the cross-domain scripting issue. See "Cross-Domain Scripting Document, Version 2.0", available at: http://www.adlnet.gov/downloads/DownloadPage.aspx?ID=58.

In order to support the training that can be provided by stand-alone applications, while at the same time providing the benefits from adherence to the SCORM standard, we have developed a Web Services version of the SCORM ECMAScript interface.

Web Services Interface

We have defined the SCORM Web Services interface with a SOAP binding. This interface is specified via Web Services Description Language (WSDL).[4] The interface is implemented as an addition to the SCORM Sample RTE, available from ADL.[5] Instructions for adding this enhancement to the SCORM Sample RTE are included at the end of this document. The Web Services interface follows the SCORM ECMAScript interface as closely as possible, while utilizing best practices for Web Services interfaces. The specifics of the Web Services interface and the differences between it and the ECMAScript interface are described below.

The WSDL defines five operations: Initialize, GetValue, SetValue, Commit, and Terminate.[6] These operations accept the same parameters and return the same results as the corresponding ECMAScript interface except as indicated below. An application's use of the Web Services methods must adhere to the same requirements as the ECMAScript interface outlined in the SCORM RTE book.[7] In particular, if the application communicates with the RTE to get and set data model elements, the application must, at a minimum, call the Initialize and Terminate methods, and it must call the Initialize method before calling other methods.

Initialization

In order to understand the differences in initialization between the ECMAScript and Web Services interfaces, it is necessary to understand the initialization process in more detail. In the ECMAScript interface, the only parameter that a SCO must supply to the Initialize method is an empty string. However, within the client side interface,[8] additional code captures the current context (for example, user ID, course ID, etc.) and attaches this to the communications session, from which the SCORM RTE can later retrieve it (see Figure 1).

[4] World Wide Web Consortium Web Services Description Language. Available at http://www.w3.org/TR/2001/NOTE-wsdl-20010315.
[5] http://www.adlnet.gov/downloads/downloadPage.aspx?id=280
[6] The WSDL may be obtained at: http://darworld.bbn.com:8080/adl/services/SCORM?wsdl
[7] SCORM 2004 3rd Edition, Run-Time Environment, ADL.
[8] Specifically, within an applet that runs on the client.

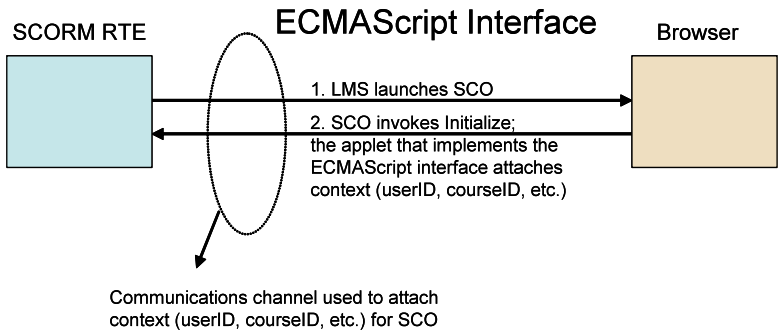

Figure 1: The Initialization process using the ECMAScript Application Programming Interface for Content to Runtime Services Communication

On receipt of the Initialize request, the SCORM Sample RTE retrieves the additional context information from the underlying communications channel between itself and the SCO in order to reference the correct data in response to subsequent requests from the SCO. While the details of how this information is associated with the SCO are not spelled out in the SCORM standard, it is clear that the association must be made in order for the RTE to access the correct data in response to requests from the SCO.

By contrast, in order to use a Web Services interface, an application only needs the URL of the Web Services endpoint. There are many ways that an application can obtain the Web services endpoint URL, including, but not limited to using an online discovery process or hard-coding the URL in the application. *The Web Services specification specifically does not cover how an application obtains the URL of the Web Services endpoint; hence this is out of scope for this document.* Having obtained the Web Services endpoint, the application initiates communication with the component that provides the Web Services interface, in this case the SCORM RTE, as illustrated in Figure 2. Note the significant difference between use of ECMAScript and Web Services interfaces: in the first case, a server launches a client and a client-side interface that collects context information for the server; in the second case, a client initiates communication with the server, which has no prior knowledge of the client.

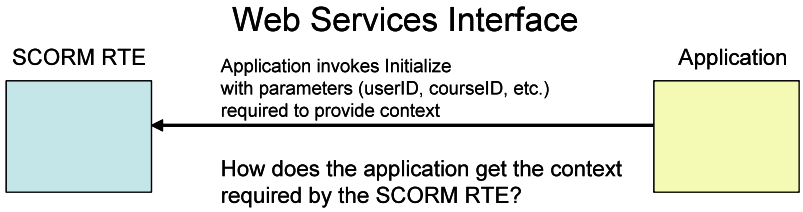

Figure 2: The Initialization process using the Web Services interface

Using the Web Services interface, the application must supply the context (user ID, course ID, etc.) to the RTE in order to enable the RTE to identify the correct data to use in response to calls to its Web Services methods. This information could be provided as parameters to each Web Services method, but for simplicity, is provided only in the Initialize method. On receipt of these parameters, the RTE associates the information with the underlying communications channel in order to correctly handle subsequent requests from the application.

The parameters to the Initialize method are: the course ID, SCO ID, user ID, user name, number of attempts, and activity ID. How these parameters are obtained by the application is outside the scope of the Web Services interface.

Error Handling
The ECMAScript interface methods: GetLastError, GetErrorString, and GetDiagnostic are not implemented in the Web Services interface. Instead, error handling in the WSDL is implemented using the SCORMException type. SCORMException encompasses three fields: scormErrorCode, scormErrorString, or scormErrorDiagnostic. The scormErrorCode is an integer, while scormErrorString and scormErrorDiagnostic are strings. The values of SCORMException are the same as the return values of the corresponding GetLastError, GetErrorString, and GetDiagnostic ECMAScript methods. The only difference is that SCORMException does not return a "No Error" (error code 0); in this case, no SCORMException is raised. A SCORMException can be thrown by the Initialize, GetValue, SetValue, Commit or Terminate WSDL operations. Receiving a SCORMException on a GetValue method is analogous to calling the GetValue method and then retrieving the error information with calls to the GetLastError, GetErrorString and GetDiagnostic methods.

In addition to SCORMExceptions, all of the WSDL methods can also throw a RemoteException. While SCORMExceptions indicate an error in following the SCORM standards as described in the SCORM RTE documentation, RemoteExceptions indicate an error in *communications* between the client and the SCORM RTE.

Browser Interface

While an application is running as described above, the Web client that is communicating with the LMS must remain open. The user should be discouraged (or prevented) from requesting additional SCOs, as the SCORM standard explicitly states that only one SCO shall be supported per user at a time. How this is achieved is outside the scope of this document. However, one possibility is that the Asset used to launch the application can also include content that can be displayed in the Web client warning the user not to request additional SCOs while the application is running. After the application has completed, the user *should* launch the next Asset or SCO using whatever controls the RTE provides to the Web client. Although the application must invoke the Terminate method in the Web Services interface when it has completed (just as any SCO must invoke the Terminate method in the ECMAScript interface), this action is not sufficient to direct the RTE to deliver the next SCO.

Installing the RTE Web Services

To install the RTE Web Services and run the SCORM Sample RTE:

1. Install the SCORM Version 2004 3rd Edition Sample Run-Time Environment Version 1.0.1, which can be obtained from ADL.[9] Note that Web Services are supported *only for this version*.

[9] Installation instructions are in the README file at:

2. Unzip the RTEWebServices.zip file and run the setup.bat file.[10] The Zip file contains all of the files required to support the RTE Web Services; the setup file copies these files into the correct locations in the SCORM Sample RTE. *It is not necessary to build the SCORM Sample Run-Time Environment.*
3. Run the SCORM Sample RTE as described in its Readme file.

After installing the RTE Web Services, and starting the SCORM Sample RTE, the WSDL can be obtained from:

http://localhost:8080/adl/services/SCORM?wsdl

Relationship to Other Standards

The "IMS General Web Services Specification"[11] recommends standards for Web Services (i.e. XML, SOAP, HTTP), guidelines for definition of the Web Services, and a process for automatically producing WSDLs from an IMS UML (Unified Modeling Language) specification, using specified IMS tools. The IMS General Web Services Specification specifies *how* to define Web Services, but it does not define a specific Web Service, i.e. the methods and data that are defined in a WSDL. One possible future task would be to compare the WSDL that we have produced with the IMS specification for producing WSDLs to determine what, if any, changes would be required to our WSDL in order to match the specification.

The "IEEE Standard for Learning Technology—Extensible Markup Language (XML) Schema Binding for Data Model for Content Object Communication"[12] is a standard for an XML binding of the IEEE 1484.11.1-2004[13] data model (commonly called the cmi data model). Our WSDL currently defines data model elements as strings or integers. A possible future task would be to define the data models using the IEEE XML encoding standard.

Acknowledgements and Contact Information

The SCORM RTE Web Services interface was developed by BBN Technologies, under contract to the Joint ADL Co-Lab as part of their 2006 Prototype effort [Contract Number: N61339-06-C-0084]. For more information, contact the BBN project manager, Virginia Travers, travers@bbn.com.

http://www.adlnet.gov/downloads/DownloadPage.aspx?ID=279
The SCORM Sample RTE is at: http://www.adlnet.gov/downloads/DownloadPage.aspx?ID=280

[10] Available from BBN on request.

[11] IMS General Web Services Specification , Version 1.0, 1/13/06. Available at: http://www.imsglobal.org/gws/index.html.

[12] IEEE Standard for Learning Technology—Extensible Markup Language (XML) Schema Binding for Data Model for Content Object Communication, IEEE Std 1484.11.3-2005. Available at http://ieeexplore.ieee.org/iel5/10875/34240/01633759.pdf?tp=&isnumber=34240&arnumber=1633759

[13] IEEE Std 1484.11.1 - 2004, IEEE Standard for Learning Technology - Data Model for Content to Learning Management System Communication, IEEE Std 1484.11.1-2004. Available at http://ieeexplore.ieee.org/xpls/abs_all.jsp?tp=&isnumber=30541&arnumber=1408444&punumber=9661

Appendix A: Web Services Interfaces

This is the Web Services Description Language (WSDL) definition for the Web Services interface described above.

```
<?xml version="1.0" encoding="UTF-8" ?>
-<wsdl:definitions targetNamespace="http://darworld.bbn.com:8080/adl/services/SCORM"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://darworld.bbn.com:8080/adl/services/SCORM"
xmlns:intf="http://darworld.bbn.com:8080/adl/services/SCORM"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:tns1="http://www.adlnet.org/xsd/adlsim_v1p3"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <!--
WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)
  -->
- <wsdl:types>
-          <schema          targetNamespace="http://www.adlnet.org/xsd/adlsim_v1p3"
xmlns="http://www.w3.org/2001/XMLSchema">
  <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
- <complexType name="SCORMException">
- <sequence>
  <element name="scormErrorCode" type="xsd:int" />
  <element name="scormErrorDiagnostic" nillable="true" type="xsd:string" />
  <element name="scormErrorString" nillable="true" type="xsd:string" />
  </sequence>
  </complexType>
  </schema>
  </wsdl:types>
- <wsdl:message name="GetValueRequest">
  <wsdl:part name="iDataModelElement" type="soapenc:string" />
  </wsdl:message>
  <wsdl:message name="CommitResponse" />
  <wsdl:message name="SetValueResponse" />
  <wsdl:message name="TerminateRequest" />
- <wsdl:message name="SetValueRequest">
  <wsdl:part name="iDataModelElement1" type="soapenc:string" />
  <wsdl:part name="iValue" type="soapenc:string" />
  </wsdl:message>
- <wsdl:message name="CommitRequest">
  <wsdl:part name="param" type="soapenc:string" />
  </wsdl:message>
- <wsdl:message name="SCORMException">
  <wsdl:part name="fault" type="tns1:SCORMException" />
  </wsdl:message>
- <wsdl:message name="InitializeRequest">
  <wsdl:part name="mCourseID" type="soapenc:string" />
  <wsdl:part name="mSCOID" type="soapenc:string" />
  <wsdl:part name="mUserID" type="soapenc:string" />
  <wsdl:part name="mUserName" type="soapenc:string" />
  <wsdl:part name="mNumAttempt" type="soapenc:string" />
```

```
  <wsdl:part name="mActivityID" type="soapenc:string" />
  </wsdl:message>
  <wsdl:message name="InitializeResponse" />
- <wsdl:message name="GetValueResponse">
  <wsdl:part name="GetValueReturn" type="soapenc:string" />
  </wsdl:message>
  <wsdl:message name="TerminateResponse" />
- <wsdl:portType name="SCORM">
- <wsdl:operation name="Terminate">
  <wsdl:input message="impl:TerminateRequest" name="TerminateRequest" />
  <wsdl:output message="impl:TerminateResponse" name="TerminateResponse" />
  <wsdl:fault message="impl:SCORMException" name="SCORMException" />
  </wsdl:operation>
- <wsdl:operation name="Initialize" parameterOrder="mCourseID mSCOID mUserID
mUserName mNumAttempt mActivityID">
  <wsdl:input message="impl:InitializeRequest" name="InitializeRequest" />
  <wsdl:output message="impl:InitializeResponse" name="InitializeResponse" />
  <wsdl:fault message="impl:SCORMException" name="SCORMException" />
  </wsdl:operation>
- <wsdl:operation name="GetValue" parameterOrder="iDataModelElement">
  <wsdl:input message="impl:GetValueRequest" name="GetValueRequest" />
  <wsdl:output message="impl:GetValueResponse" name="GetValueResponse" />
  <wsdl:fault message="impl:SCORMException" name="SCORMException" />
  </wsdl:operation>
- <wsdl:operation name="SetValue" parameterOrder="iDataModelElement1 iValue">
  <wsdl:input message="impl:SetValueRequest" name="SetValueRequest" />
  <wsdl:output message="impl:SetValueResponse" name="SetValueResponse" />
  <wsdl:fault message="impl:SCORMException" name="SCORMException" />
  </wsdl:operation>
- <wsdl:operation name="Commit" parameterOrder="param">
  <wsdl:input message="impl:CommitRequest" name="CommitRequest" />
  <wsdl:output message="impl:CommitResponse" name="CommitResponse" />
  <wsdl:fault message="impl:SCORMException" name="SCORMException" />
  </wsdl:operation>
  </wsdl:portType>
- <wsdl:binding name="SCORMSoapBinding" type="impl:SCORM">
  <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
- <wsdl:operation name="Terminate">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="TerminateRequest">
        <wsdlsoap:body       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://server.samplerte.adl.org" use="encoded" />
  </wsdl:input>
- <wsdl:output name="TerminateResponse">
        <wsdlsoap:body       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://darworld.bbn.com:8080/adl/services/SCORM" use="encoded" />
  </wsdl:output>
- <wsdl:fault name="SCORMException">
       <wsdlsoap:fault       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="SCORMException"       namespace="http://darworld.bbn.com:8080/adl/services/SCORM"
use="encoded" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="Initialize">
  <wsdlsoap:operation soapAction="" />
```

```
- <wsdl:input name="InitializeRequest">
        <wsdlsoap:body       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://server.samplerte.adl.org" use="encoded" />
  </wsdl:input>
- <wsdl:output name="InitializeResponse">
        <wsdlsoap:body       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://darworld.bbn.com:8080/adl/services/SCORM" use="encoded" />
  </wsdl:output>
- <wsdl:fault name="SCORMException">
       <wsdlsoap:fault       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="SCORMException"        namespace="http://darworld.bbn.com:8080/adl/services/SCORM"
use="encoded" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="GetValue">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="GetValueRequest">
        <wsdlsoap:body       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://server.samplerte.adl.org" use="encoded" />
  </wsdl:input>
- <wsdl:output name="GetValueResponse">
        <wsdlsoap:body       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://darworld.bbn.com:8080/adl/services/SCORM" use="encoded" />
  </wsdl:output>
- <wsdl:fault name="SCORMException">
       <wsdlsoap:fault       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="SCORMException"        namespace="http://darworld.bbn.com:8080/adl/services/SCORM"
use="encoded" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="SetValue">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="SetValueRequest">
        <wsdlsoap:body       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://server.samplerte.adl.org" use="encoded" />
  </wsdl:input>
- <wsdl:output name="SetValueResponse">
        <wsdlsoap:body       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://darworld.bbn.com:8080/adl/services/SCORM" use="encoded" />
  </wsdl:output>
- <wsdl:fault name="SCORMException">
       <wsdlsoap:fault       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="SCORMException"        namespace="http://darworld.bbn.com:8080/adl/services/SCORM"
use="encoded" />
  </wsdl:fault>
  </wsdl:operation>
- <wsdl:operation name="Commit">
  <wsdlsoap:operation soapAction="" />
- <wsdl:input name="CommitRequest">
        <wsdlsoap:body       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://server.samplerte.adl.org" use="encoded" />
  </wsdl:input>
- <wsdl:output name="CommitResponse">
        <wsdlsoap:body       encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://darworld.bbn.com:8080/adl/services/SCORM" use="encoded" />
```

```
  </wsdl:output>
- <wsdl:fault name="SCORMException">
       <wsdlsoap:fault      encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="SCORMException"        namespace="http://darworld.bbn.com:8080/adl/services/SCORM"
use="encoded" />
  </wsdl:fault>
  </wsdl:operation>
  </wsdl:binding>
- <wsdl:service name="SCORMService">
- <wsdl:port binding="impl:SCORMSoapBinding" name="SCORM">
  <wsdlsoap:address location="http://darworld.bbn.com:8080/adl/services/SCORM" />
  </wsdl:port>
  </wsdl:service>
  </wsdl:definitions>
```